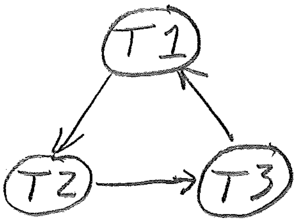
David Kalish
ps3_ part I

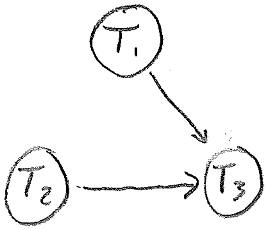
① Schedule 1:

$w_1(A)$; $r_2(A)$; $w_2(B)$;
$r_3(B)$; $w_3(C)$; $r_1(C)$



cyclic: not conflict serializable

Schedule 2: $r_1(C)$; $w_1(A)$; $r_3(A)$; $w_2(B)$; $r_3(B)$; $w_3(C)$



acyclic: conflict serializable

Schedule 1:

T_1	T_2	T_3
w(A)		
	r(A)	
	w(B)	
		r(B)
		w(C)
r(C)		

≠

T_1	T_2	T_3
w(A)		
r(C)		
	r(A)	
	w(B)	
		r(B)
		w(C)

not equivalent because T_1's read of C needs to come after T_3's write of C. Now it comes before.

Schedule 2:

T_1	T_2	T_3
r(C)		
w(A)		
		r(A)
	w(B)	
		r(B)
		w(C)

=

T_1	T_2	T_3
r(C)		
w(A)		
	w(B)	
		r(A)
		r(B)
		w(C)

equivalent because in both versions, T_3's read of A comes after T_1's write of A, and T_3's read of B comes after T_2's write of B

② 2-phase locking

T1: r(B) r(C) r(A) w(C) c

T2: r(B) r(A) r(C) w(A) c

T1	T2
sl(B)	
r(B)	
	sl(B)
	r(B)
	xl(A)
	sl(C)
	u(B)
sl(C)	
r(C)	
	r(A)
	r(C)
	w(A)
	c
	u(A)
	u(C)
sl(A)	
xl(C)	
r(A)	
w(C)	
c	
u(A)	
u(B)	
u(C)	

2) not rigorous because T2 unlocks B early

3) Not recoverable because T1 does writes after T2's commit. If system crashes after $w_1(C)$ but befor T1 commits, T2 will be restored while T1 undoes changes.

4) recoverable version:

T1	T2
sl(B)	
r(B)	
	sl(B)
	r(B)
	xl(A)
	sl(C)
	u(B)
sl(C)	
r(C)	
	r(A)
	r(C)
	w(A)
	u(A)
	u(C)
sl(A)	
xl(C)	
r(A)	
w(C)	
u(A)	
u(B)	
u(C)	
	c
c	

5) not cascadeless because T1's r(A) is a dirty read after T2 writes A

③ Lock Modes

$xl_2(A)$ $w_2(A)$ $sl_1(B)$ $r_1(B)$ $ul_3(C)$ ...

1) $ul_1(A)$ - no, A is under exclusive lock in T_2

2) $sl_3(B)$ - yes, B is under shared lock in T_1

3) $ul_2(B)$ - yes, B is shared lock in T_1

4) $sl_1(C)$ - no, C is update-locked in T_3

5) $xl_3(C)$ - yes, this upgrades T_3's update lock

6) $ul_2(C)$ - no, T_3 has C update locked, only 1 T can have an update lock of a given item at a time.

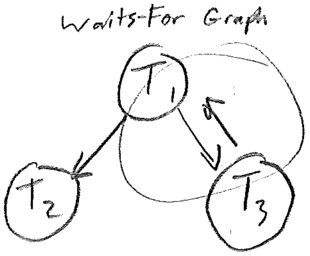
④ Deadlock Detection

seq 1: $r_2(C)$; $r_1(D)$; $w_1(C)$; $r_3(C)$; $w_2(C)$; $w_3(D)$; $w_1(D)$

T_1	T_2	T_3
	sl(C); r(C)	
sl(D); r(D)		
xl(C) DENIED wait for T_2		sl(C); r(C)

	xl(C); w(C)
	commit
	u(C)	

xl(C) DENIED wait for T_3		
		xl(D) DENIED wait for T_1



cycle:
DEADLOCK

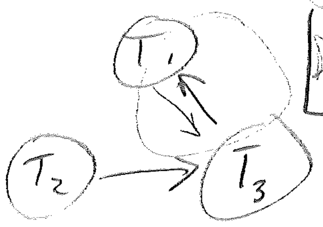
④ Dedlck Detection - Sequence 2

$w_3(A)$; $r_2(A)$; $w_2(A)$; $r_1(B)$; $r_3(B)$; $w_3(B)$; $w_1(B)$

T_1	T_2	T_3
		xl(A); w(A)
	sl(A) Denied wait for T_3	
sl(B); r(B)		

		sl(B); r(B)
		xl(B)
		DENIED
		wait for T_1

xl(B) DENIED wait for T_3		



DEADLOCK

⑤ Timestamps & Mult Vers.

① $TS_1 = 100$; $TS_2 = 101$; $TS_3 = 102$; $TS_4 = 103$; $TS_5 = 104$

$w_3(A)$ - not allowed because T_5 is most recent timestamp
Abort T_3

$w_1(A)$ - Not allowed, T_5 is most recent timestamp
Abort T_1

$W_4(A)$ - Not allowed because T_5 is most recent timestamp
Abort T_4

$r_5(A)$ - Allowed because T_5 is most recent timestamp
$RTS(A) = 105$

$r_2(A)$ - Not allowed, T_5 most recent timestamp
Abort T_2

C_1, C_2, C_3, C_4 - ignored because they have been aborted

C_5 - Commit T_5 allowed because most recent TS is in T_5

⑤ 2) with commit bit

S_1 - TS_1 = 100

S_2 - TS_2 = 101

S_3 - TS_3 = 102

S_4 - TS_4 = 103

S_5 - TS_5 = 104

$W_3(A)$ - Not allowed, must wait for T_5 & T_4 to commit

$W_1(A)$ - Not allowed, must wait for $T_{2,3,4,5}$ to commit

$W_4(A)$ - Not allowed, must wait for T_5 to commit

$r_5(A)$ - Allowed because T_5 is current TS.

RTS(A) = 105 , A. commit = false

$r_2(A)$ - Notallowed, must wait for $T_{3,4,5}$ to commit

$C_{1,2,3,4}$ - ignored, wait for C_5

C_5 - commit, retry T_4 actions A.commit = true

$w_4(A)$ - Allowed because T_5 committed. A.commit = false

WTS(A) = 106

C_4 - commitT_4, retry T_3 actions. A. commit = True

$W_3(A)$ - Allowed because $T_{4,5}$ committed. A.commit = false

WTS(A) = 107

C_3 - Commit T_3, retry T_2. A.commit = True

$r_2(A)$ - Allowed because $T_{3,4,5}$ committed. A.commit = False

RTS(A) = 108

C_2 - commit T_2, retry T_1. A. commit = true

$W_1(A)$ - Allowed, $T_{2,3,4,5}$ committed. A.commit = false

WTS(A) = 109

C_1 - commit T_1. All transactions committed & finished

⑤ 3) multiversion TS

S_1, S_2, S_3, S_4, S_5 - $TS_1 = 100$, $TS_2 = 105$, $TS_3 = 110$, $TS_4 = 115$, $TS_5 = 120$

$w_3(A)$ - write $A(110)$. $WTS(A_{110}) = 125$

$w_1(A)$ - write $A(100)$. $WTS(A_{100}) = 130$

$w_4(A)$ - write $A(115)$. $WTS(A_{115}) = 135$

$r_5(A)$ - read $A(120)$. $RTS(A_{120}) = 140$

$r_2(A)$ - read $A(105)$. $RTS(A_{105}) = 145$

c_1 - commit T_1 with $A(130)$

c_2 - commit T_2 with $A(145)$

c_3 - commit T_3 with $A(125)$

c_4 - commit T_4 with $A(135)$

c_5 - commit T_5 with $A(140)$